Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Curso de Ciência da Computação

VINICIUS ROGÉRIO ARAUJO SILVA

RECORTE DE DADOS VOLUMÉTRICOS USANDO WIIMOTE E KINECT

São Luís
2012

VINICIUS ROGÉRIO ARAUJO SILVA

RECORTE DE DADOS VOLUMÉTRICOS USANDO WIIMOTE E KINECT

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal do Maranhão, para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Anselmo Cardoso de Paiva

São Luís

2012

Silva, Vinicius Rogério Araujo

Recorte de Dados Volumétricos Usando WiiMote e Kinect / Vinicius Rogério Araujo Silva. – 2012.

41f.

Impresso por computador (Fotocópia).

Orientador: Anselmo Cardoso de Paiva.

Monografia (graduação) – Universidade Federal do Maranhão, Curso de Ciência da Computação, 2012.

1.Computação gráfica 2. Imagens médicas 3. Dados volumétricos I.Título.

CDU 004.92:61

VINICIUS ROGÉRIO ARAUJO SILVA

RECORTE DE DADOS VOLUMÉTRICOS USANDO WIIMOTE E KINECT

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal do Maranhão, para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em: 14/09/2012

BANCA EXAMINADORA

Prof. Anselmo Cardoso de Paiva (Orientador)

Doutor em Informática

Universidade Federal do Maranhão

Prof. Aristófanes Corrêa Silva

Doutor em Informática

Universidade Federal do Maranhão

Prof. Simara Vieira da Rocha

Mestre em Egenharia Elétrica

Universidade Federal do Maranhão

RESUMO

Os enormes avanços no campo da visualização científica têm permitido o desenvolvimento de inovadoras aplicações de visualização destes dados. Neste contexto, um dos grandes destaques é a visualização de imagens médicas, onde representações tridimensionais precisas do corpo humano e suas estruturas internas podem ser geradas. Com o uso de imagens médicas e o potencial de visualização das representações tridimensionais de dados médicos, surgiram ferramentas capazes de manipular os volumes provenientes destes dados de forma que suas estruturas internas pudessem ser visualizadas. Este trabalho apresenta maneiras diferentes de interação que permitem a um usuário controlar a visualização das estruturas internas de imagens médicas tridimensionais em um ambiente de visualização, através de dois dispositivos de entrada que fogem dos métodos mais tradicionais: o WiiMote e o Kinect.

Palavras-chave: Computação gráfica, Imagens médicas, Dados volumétricos.

ABSTRACT

The huge advances in the field of scientific visualization has allowed for the development of inovative data visualization applications. In this context, one of the great highlights is the visualization of medical images, where accurate tridimentional representations of the human body and its internal structures can be generated. By using medical images and with the visualization potential allowed for the tridimentional representation of medical data, new tools have been developed, allowing for the manipulation of the volumes generated through this data, in a way that its internal structures could be visualized. This work presents different ways of interaction, which allow the user to control the visualization of internal structures of tridimentional medical images in a visualization ambient, by using two input devices that drift away from the traditional methods: WiiMote and Kinect.

Keywords: Computer graphics, Medical images, Volumetric data.

Sumário

1 INTRODUÇÃO 5

2 FUNDAMENTAÇÃO TEÓRICA 7

2.1 Visualização de dados médicos 7

2.1.1 Dados volumétricos 8

2.1.2 Técnicas de visualização volumétrica 9

2.1.3 Ferramentas de recorte 10

2.2 Acelerômetros 12

2.2.1 WiiMote 13

2.3 Kinect 14

3 PROPOSTA 18

3.1 Ferramentas utilizadas 18

3.2 Detalhes da implementação e arquitetura 20

3.3 Uso da aplicação 23

4 CONCLUSÃO 34

REFERÊNCIAS 36

Lista de Figuras

2.1 Visualização 2D de dados médicos através de fatias. 7
2.2 Visualização 3D de dados médicos: visualização volumétrica. 8
2.3 Representação de um voxel (PAIVA; SEIXAS; GATTASS, 1999). 9
2.4 Controle *WiiMote* e *Nintendo Wii* (HOW..., 2012). 14
2.5 Sensor *Kinect* (KINECT..., 2012). 14
2.6 Pontos de rastreamento do *Kinect* no corpo do usuário (KINECT..., 2012). 15
2.7 Hardware do sensor *Kinect* (KINECT..., 2012). 16

3.1 Métodos principais e hierarquia da classe CInputDeviceProcessor. 21
3.2 Métodos principais e hierarquia da classe CClippingAlgorithm. 22
3.3 Aplicação desenvolvida em execução. 23
3.4 Uso da ferramenta de recorte Borracha 2D. 25
3.5 Uso da ferramenta de recorte Guilhotina 2D. 26
3.6 Uso da ferramenta de Recorte Poligonal 2D. 28
3.7 Uso da ferramenta de recorte Borracha 3D. 29
3.8 Uso da ferramenta de recorte Guilhotina 3D. 29
3.9 Eixos do *WiiMote*. 30
3.10 Painel de configurações e monitoramento do *WiiMote*. 31
3.11 Painel de configurações e utilização do *Kinect*. 32

1 INTRODUÇÃO

No domínio da computação gráfica, a visualização diz respeito à geração de imagens a partir de conjuntos de dados, com o objetivo de realizar uma interpretação acerca destes dados (MCCORMICK; DEFANTI; BROWN, 1987). Sua utilização se dá nas mais diversas situações, desde as mais simples aplicações de uso diário até as mais complexas, como jogos eletrônicos, aplicações para manipulação de imagens, simulações, modelagem de projetos mecânicos, construção civil, dentre inúmeros outros exemplos.

A visualização possui diversas subáreas, dentre as quais podemos destacar a visualização científica, que trabalha com conjuntos de dados complexos para a extração de informações científicas relevantes. A visualização científica trata principalmente da visualização de dados resultantes de medições e simulação de fenômenos reais. Como exemplos de aplicações que se utilizam da visualização científica, podemos citar aplicações de simulação meteorológica, de simulação de fluidos, de análise geológica e aplicações médicas. Dentro da visualização científica, por diversas vezes faz-se necessário analisar conjuntos de dados que representam regiões volumétricas. Neste caso, a visualização passa a ser denominada visualização volumétrica.

Recentemente, a visualização volumétrica vem sendo aprimorada para ser utilizada na visualização de dados médicos. A partir do uso de dados obtidos através de exames médicos, tais como a Tomografia Computadorizada e a Ressonância Magnética, e do uso de aplicações de *rendering*, podem ser construídos modelos tridimensionais que imitam uma representação real de um objeto de estudo — em geral, o corpo de um paciente. Isto permite o desenvolvimento de aplicações médicas para diversas finalidades, tais como auxiliar no diagnóstico clínico do paciente, realizar o planejamento de uma cirurgia ou explorar o corpo humano sem a utilização de métodos invasivos.

Uma das características mais importantes para aplicações de visualização de dados complexos é a maneira como o usuário interage com os dados que estão sendo visualizados. Por isso, é importante explorar alternativas aos dispositivos de entrada mais tradicionais (*mouse* e teclado) utilizados por usuários nos computadores onde a visualização de dados é feita. O uso de dispositivos de entrada mais intuitivos podem fazer com que o usuário se sinta mais confortável e aumentar o grau de imersão em relação à aplicação, gerando resultados positivos na interação entre o usuário e a aplicação.

O objetivo deste trabalho foi o desenvolvimento de um ambiente de visualização de dados volumétricos que oferecesse suporte ao uso de dispositivos de entrada que pudessem ser mais intuitivos para o usuário, facilitando a interação com a aplicação. Mais especificamente, foi desenvolvido um ambiente de visualização de dados obtidos através de exames médicos, capaz de realizar a exploração das regiões internas do volume gerado por estes dados e que suporta o uso de dois dispositivos de entrada que surgiram recentemente na industria de jogos eletrônicos: o *WiiMote* e o *Kinect*.

Este trabalho está organizado em quatro capítulos. Iniciaremos o Capítulo 2 discutindo os principais conceitos utilizados durante o desenvolvimento deste projeto e fornecendo o embasamento necessário para a compreensão da aplicação desenvolvida.

No Capítulo 3, a aplicação desenvolvida será descrita, assim como os principais detalhes acerca de sua implementação e de sua utilização por parte do usuário.

Finalmente, no Capítulo 4 apresentamos as considerações finais do projeto e algumas possibilidades futuras para melhoria de suas funcionalidades.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Visualização de dados médicos

O termo “Visualização” está relacionado aos métodos que permitem a extração de informações relevantes a partir de conjuntos complexos de dados (PAIVA; SEIXAS; GATTASS, 1999). Este processo geralmente é feito através da utilização de técnicas de computação gráfica e processamento de imagens. Segundo (MCCORMICK; DEFANTI; BROWN, 1987), a visualização é uma ferramenta para a interpretaçao de conjuntos de dados e geração de imagens a partir destes dados, os quais são, em geral, complexos e multidimensionais. Quando esses conjuntos de dados representam fenômenos complexos e o objetivo é a extração de informações científicas relevantes, a visualização passa a ser classificada como visualização científica (MCCORMICK; DEFANTI; BROWN, 1987).

A visualização de dados médicos pode ser feita de duas maneiras distintas. A primeira, mais tradicional, é através da utilização de imagens bidimensionais, denominadas fatias. A Figura 2.1 ilustra a visualização de fatias obtidas através de um exame de Tomografia Computadorizada. Um ponto positivo da visualização de dados médicos através de fatias é que a visualização de imagens bidimensionais necessita de poucos recursos computacionais. Porém, analise dessas fatias requer, em geral, médicos especialistas capazes de detectar adversidades presentes nos exames e um treinamento por parte destes especialistas.

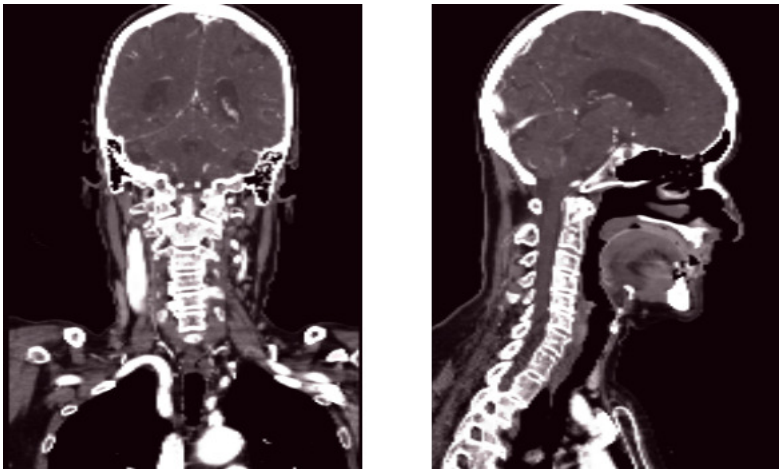

Figura 2.1: Visualização 2D de dados médicos através de fatias.

A segunda forma de visualizar dados médicos dá-se através da Visualização Vo-

lumétrica. A visualização volumétrica é uma subárea da visualizaćão científica, e agrega um conjunto de técnicas utilizadas na visualização de dados das regiões de um volume. Estes dados associados às regiões de um volume são denominados dados volumétricos. A visualização volumétrica foi a forma de visualização utilizada neste trabalho, e será aprofundada nas Seções 2.1.2 e 2.1.3. A Figura 2.2 representa a visualização volumétrica de dados médicos obtidos através de um exame de Tomografia Computadorizada. Esta forma de visualizar exames médicos possui uma menor necessidade de ter um especialista que seja capaz de interpretar as imagens, como é o caso da visualização através de fatias.

Figura 2.2: Visualização 3D de dados médicos: visualização volumétrica.

2.1.1 Dados volumétricos

Os conjuntos de dados volumétricos são, em geral, definidos em grades tridimensionais, tratados como matrizes tridimensionais de elementos de volume. Esses elementos de volume são chamados de *voxels* (*Volume Picture Elements*, ou ainda *Volumetric Pixel*). *Voxels* são paralelepípedos, fortemente agrupados, formados pela divisão do espaço do objeto através de um conjunto de planos paralelos aos eixos principais desse espaço. Esses elementos não devem se interceptar, sendo de tamanho suficientemente pequeno se comparado às características representadas pelos dados volumétricos (SOUZA, 2010; HUFF, 2006). Em uma analogia com o conceito de *pixel*, podemos referenciar o *voxel* como sendo a tupla $\langle i, j, k, S \rangle$ que define um ponto amostrado do campo escalar na posição (i, j, k) e com valor S associado.

Levando em consideração a origem dos dados volumétricos, podemos classificá-los como provenientes de duas fontes principais:

Simulação Os dados são construídos a partir de um modelo matemático.

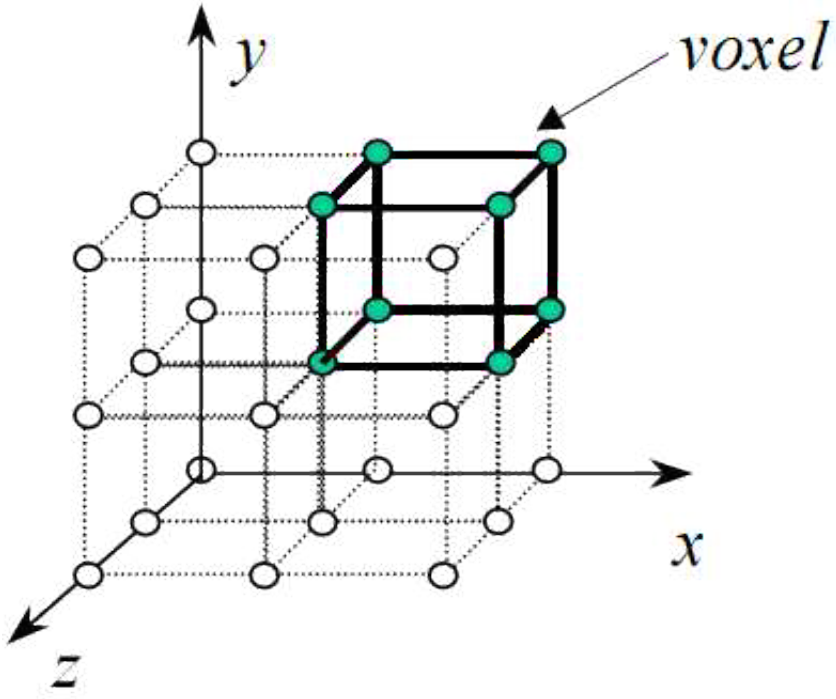

Figura 2.3: Representação de um voxel (PAIVA; SEIXAS; GATTASS, 1999).

Aquisição Os dados são adquiridos a partir da conversão de objetos existentes na natureza, geralmente através do uso de *scanners* sobre o material de interesse. Podemos citar como exemplos de técnicas de aquisição a Ressonância Magnética, a Tomografia Computadorizada, Sismógrafos, dentre outros.

O foco deste trabalho, em termos de dados volumétricos, serão os dados médicos obtidos através de processos de aquisição em exames de Ressonância Magnética e Tomografia Computadorizada, que geram uma série de fatias bidimensionais (*slices*), a partir das quais é possível construir um volume tridimensional que será visualizado pelo usuário.

2.1.2 Técnicas de visualização volumétrica

A visualização volumétrica, em termos gerais, é um processo de *rendering*, cujo objetivo é obter a melhor compreensão da estrutura armazenada nos dados volumétricos (MANSSOUR et al., 2002). Existem duas abordagens básicas para a realização deste processo: os algoritmos de *rendering* direto de volumes e os algoritmos de extração ou adaptação de superfícies (*surface-fitting algorithms*) (ELVINS, 1992). A diferença entre as duas abordagens está na utilização ou não de representações intermediárias dos dados volumétricos para a geração da visualização.

As técnicas de *rendering* direto de volumes realizam a projeção diretamente a partir

dos dados volumétricos, enquanto as técnicas de extração de superfícies geram uma representação geométrica a partir dos dados volumétricos, permitindo o uso dos métodos tradicionais de *rendering* de polígonos para a geração da visualização.

Estas técnicas envolvidas no processo de visualização podem ser simplificadas em quatro passos fundamentais:

1. **Formação do Volume:** Envolve a aquisição dos dados, o pré-processamento das fatias e a reconstrução do volume;
2. **Classificação:** Relacionada à identificação do material representado em cada *voxel*, permitindo a seleção de características dos dados, segundo um critério quantitativo, definindo a região do volume que se deseja explorar;
3. **Iluminação:** Cálculo da tonalidade de cor em cada ponto do volume, com base nas propriedades do material de cada *voxel* e condições de iluminação externa;
4. **Projeção:** Ocorre a projeção dos *voxels*, ou polígonos mapeados, na superfície de visualização e a consequente composição para determinar a imagem a ser visualizada.

Neste trabalho, o *rendering* direto foi escolhido como a técnica de visualização a ser utilizada, pois esta técnica utiliza os dados volumétricos diretamente, sem a necessidade de representações intermediárias. Isto permite a implementação de Técnicas de Recorte (Seção 2.1.3) sobre o volume através da eliminação de dados que não são interessantes ao usuário.

Uma etapa que merece destaque, por sua importância para este trabalho, é a etapa de Classificação. Nas técnicas de extração de superfícies, durante esta etapa, é definido o valor de limiarização (*threshold*) utilizado para a identificação da superfície a ser transformada em polígonos. Já nas técnicas baseadas em *rendering* direto, durante a etapa de Classificação, são definidas as relações entre os valores do volume (o valor associado a cada *voxel* que compõe o volume) e suas cores e opacidades. Estas relações são denominadas funções de transferência.

2.1.3 Ferramentas de recorte

Os conjuntos de dados volumétricos em algumas situações tendem a ser bastante complexos. Para dados deste tipo, muitas vezes é necessário permitir que o usuário tenha

recursos para visualizar as estruturas internas do volume, ou isolar regiões de interesse no mesmo. Podemos citar, como exemplo prático, o uso de aplicações médicas para explorar os órgãos internos de um paciente, onde o médico deve se utilizar destes recursos para remover os tecidos externos (pele, gordura, etc.) da visualização e manter apenas as regiões de interesse do volume (neste exemplo, as regiões correspondentes aos órgãos internos do paciente).

Esta tarefa de remoção das regiões não interessantes do volume implica na seleção dos *voxels* que serão removidos da visualização. Após a seleção, estes *voxels* podem ser removidos, por exemplo, através da alteração dos seus valores de opacidade, tornando-os "transparentes" durante o processo de *rendering*.

Nesse contexto, podem ser usadas ferramentas que permitam que um usuário selecione quais *voxels* do volume são irrelevantes e, portanto, podem ser excluídos da visualização dos dados. Essas ferramentas são denominadas Ferramentas de Recorte, também conhecidas como Ferramentas de Escultura, pois ao utilizá-las, o usuário está, efetivamente, esculpindo o volume tridimensional e mantendo apenas o que for considerado interessante durante este processo.

As ferramentas de recorte utilizadas neste trabalho são formas geométricas bidimensionais e tridimensionais (os *cursores*), sobre as quais são executados testes de detecção de interseção com os *voxels* do volume que está sendo visualizado pelo usuário. Para os cursores bidimensionais, os testes de interseção são realizados entre o cursor, que se encontra no plano de visualização, e a projeção de cada *voxel* neste plano. Já para os cursores tridimensionais, o usuário é capaz de definir a posição e rotacionar o cursor (que será uma geometria no espaço tridimensional) na cena, e os testes de interseção são realizados entre os pontos que representam a posição dos *voxels* na cena e o cursor.

A partir desses testes de interseção, a ferramenta pode determinar quais *voxels* serão selecionados e removidos da visualização. Algumas das ferramentas são utilizadas para remover os *voxels* que estão em interseção com o cursor, enquanto outras são utilizadas para remover os *voxels* que não estão em interseção com o cursor, mantendo apenas os voxels que apresentam interseção.

Neste trabalho, foram implementadas cinco ferramentas de recorte, denominadas: *Borracha 2D*, *Borracha 3D*, *Guilhotina 2D*, *Guilhotina 3D* e *Recorte Poligonal 2D*. Todas as ferramentas que usam cursores bidimensionais (ferramentas 2D) são controladas pelo usuário através da utilização de dispositivos de entrada tradicionais: *mouse* e teclado. Já

as ferramentas baseadas em cursores tridimensionais têm suporte aos dispositivos *Kinect* e *WiiMote*.

As ferramentas Borracha 2D e Borracha 3D são baseadas em cursores cilíndricos. Os *voxels* selecionados por estas ferramentas são aqueles que possuem interseção com o cursor. No caso da Borracha 2D, o cursor é um cilindro cujo eixo é ortogonal ao plano de visualização. Para a Borracha 3D, o cursor é um cilindro, cujo posicionamento e a orientação podem ser controlados pelo usuário.

As ferramentas de Guilhotina 2D e Guilhotina 3D possuem um plano como cursor. Os *voxels* selecionados por estas ferramentas são aqueles que possuem interseção com um dos semiespaços divididos pelo plano. Na ferramenta de Guilhotina 2D, o cursor plano é obrigatoriamente perpendicular ao plano de visualização. Já na ferramenta de Guilhotina 3D, o posicionamento e a orientação do plano podem ser controlados pelo usuário.

Por último, a ferramenta de Recorte Poligonal 2D é uma ferramenta que permite ao usuário definir polígonos convexos arbitrários e selecionar a região interna ou a região externa do polígono, para que todos os *voxels* pertencentes à região selecionada sejam removidos da visualização. Por ser uma ferramenta bidimensional, vale lembrar que isto é feito através da projeção dos voxels no plano de visualização e o teste é feito através de um algoritmo de detecção de interseção com polígonos em um plano de duas dimensões.

2.2 Acelerômetros

Acelerômetros são dispositivos eletromecânicos que funcionam como sensores capazes de medir forças de aceleração. Estas forças podem ser classificadas como estáticas, como a força constante da gravidade, ou dinâmicas, causadas pela movimentação ou vibração do acelerômetro (ANDREJASIC; POBERAJ, 2008). Um acelerômetro triaxial é um sensor que retorna valores estimados de aceleração ao longo dos eixos X, Y e Z. Estes valores podem ser utilizados para estimar velocidade, deslocamento e orientação (RAVI et al., 2005).

Até alguns anos atrás, os custos de produção dos acelerômetros eram muito altos e sua utilização estava condicionada a áreas bastante específicas, onde seu custo era irrelevante, tais como em sistemas militares e aeroespaciais. Contudo, os custos vêm se tornando cada vez mais suportáveis, de forma que a utilização dos acelerômetros passou a ser menos limitada (FIGUEIREDO et al., 2007).

Além disso, avanços na miniaturização destes sensores permitiram que eles fossem embutidos nos mais variados dispositivos, tais como telefones celulares, controles para jogos eletrônicos, sistemas de airbag, sistemas de detecção de vibrações para máquinas e componentes eletrônicos, dentre outros, a um custo acessível (ANDREJASIC; POBERAJ, 2008; RAVI et al., 2005).

2.2.1 WiiMote

Neste trabalho, utilizamos um dispositivo de entrada que possui um acelerômetro triaxial embutido e que é capaz de consultar este acelerômetro e repassar seus valores de aceleração diretamente para o aplicativo desenvolvido. Trata-se de um controle para jogos eletrônicos chamado *WiiMote*, também conhecido por *Wii Remote* ou *Wii Controller*.

O *WiiMote* é o controle primário do console *Nintendo Wii*. Além do acelerômetro embutido, o *WiiMote* também possui outros recursos, alguns dos quais foram explorados pela aplicação desenvolvida neste trabalho, tais como botões (12, no total) e uma porta onde podem ser encaixados acessórios para acrescentar funcionalidades ao dispositivo. O *WiiMote* também possui três recursos simples de *output* de dados: ele é capaz de vibrar, pode emitir sons através de um alto-falante embutido e possui 4 LEDs que podem ser ligados ou desligados arbitrariamente. O controle também possui uma câmera capaz de captar luz infravermelha. O *Nintendo Wii* vem com uma barra horizontal que possui dois pontos de emissão de luz infravermelha, que podem ser monitorados por esta câmera do *WiiMote*. Através do processamento da posição das luzes infravermelhas no campo de visão da câmera embutida no *WiiMote*, o controle é capaz de ser usado como um dispositivo apontador pelo usuário.

A Figura 2.4 ilustra o *WiiMote*, ao lado do console *Nintendo Wii*.

A facilidade de uso deste controle o tornou bastante popular na *web*, de forma que vários *websites* não-oficiais fornecem informações tecnicas bastante precisas sobre ele, obtidas através de processos de engenharia reversa (WIIBREW, 2012; WIILI, 2012; HOW. . . , 2012).

A comunicação do *WiiMote* se dá através de um *link Bluetooth*. Ele é capaz de enviar atualizações a uma frequência máxima de 100 atualizações por segundo, o que permite o seu uso satisfatório em aplicações interativas. O *WiiMote* não requer autenticação para o estabelecimento de conexões com um *host*.

Figura 2.4: Controle *WiiMote* e *Nintendo Wii* (HOW..., 2012).

A utilização do *WiiMote* neste trabalho será aprofundada no Capítulo 3.

2.3 Kinect

O *Kinect* é um controle de jogos eletrônicos apresentado em novembro de 2010 pela Microsoft. Trata-se de um dispositivo capaz de interpretar cenas 3D a partir de uma estrutura baseada em câmeras, permitindo que o usuário interaja com um mundo virtual através de uma interface natural (NUI, *Natural User Interface*). Denominam-se Interfaces Naturais de Usuário as interfaces de usuário que são transparentes ao utilizador, de forma que as interações nesse tipo de interface ocorrem através de ações intuitivas relacionadas ao comportamento natural humano. Em junho de 2011, a Microsoft liberou o pacote de desenvolvimento de *software* oficial do *Kinect* (*Kinect SDK*) para o sistema operacional Windows 7.

Figura 2.5: Sensor *Kinect* (KINECT..., 2012).

Dentre os recursos oferecidos pelo *Kinect*, o principal é a sua capacidade de identificar

e rastrear a posição de usuários através de seu sistema de câmeras. O *Kinect* é capaz de identificar e rastrear a posição de até seis usuários diferentes. Além disso, o *Kinect* possui um recurso chamado "*Skeletal Tracking*", que consiste no rastreamento e estimativa do posicionamento de determinados pontos do corpo do usuário. Este recurso é limitado ao rastreamento dos pontos do corpo de apenas dois usuários simultaneamente. No total, o *Kinect* é capaz de rastrear vinte pontos diferentes do corpo de um usuário, que são: a cabeça, as mãos, os pulsos, os cotovelos, o centro dos ombros (abaixo do pescoço), os ombros, a coluna, o centro dos quadris (na região lombar), os quadris, os joelhos, os calcanhares e os pés. A Figura 2.6 ilustra todos os pontos do corpo do usuário que o *Kinect* é capaz de rastrear.

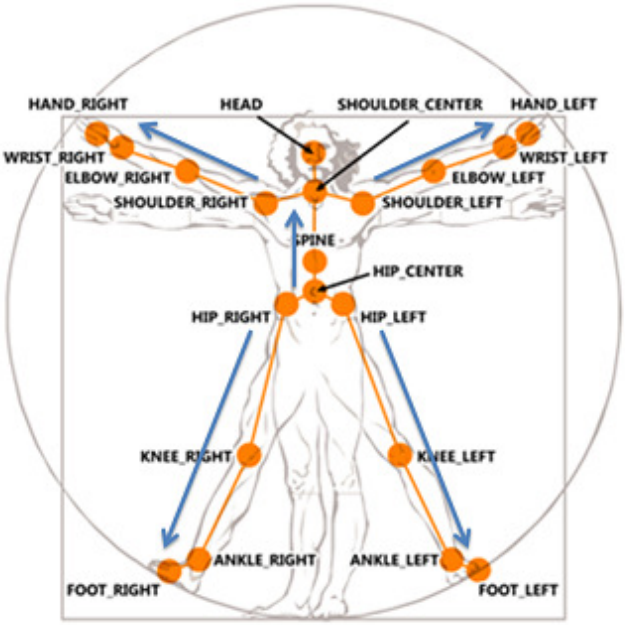

Figura 2.6: Pontos de rastreamento do *Kinect* no corpo do usuário (KINECT..., 2012).

Basicamente, a nível de *hardware* o *Kinect* é constituído de uma câmera RGB, um sensor de profundidade, uma matriz de microfones para captação de áudio e um motor. O motor é utilizado para modificar o ângulo de elevação do *Kinect*, e pode alterar a inclinação do sensor dentro de uma variação de $\pm 27°$. O uso de múltiplos microfones permite a captura de som e o processamento do áudio: o *Kinect* oferece recursos de supressão de ruídos, cancelamento de eco acústico, localização da fonte de um som e reconhecimento de voz. Todos esses recursos de áudio são possíveis graças à organização dos microfones dentro do sensor. A câmera RGB é usada em conjunto com o sensor de profundidade durante o processo de rastreamento dos usuários. O sensor de profundidade consiste em um projetor de laser infravermelho combinado com um sensor monocromático CMOS (*Complementary metal-oxide-semiconductor*). A Figura 2.7 ilustra os principais

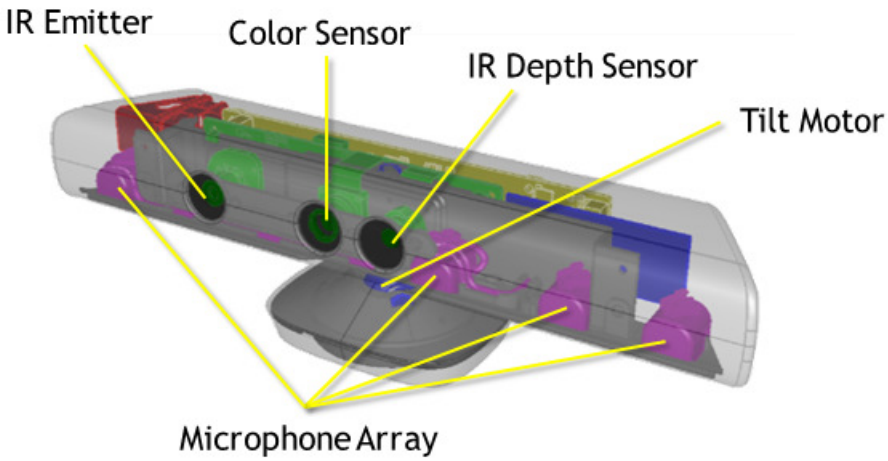

Figura 2.7: Hardware do sensor *Kinect* (KINECT..., 2012).

componentes de *hardware* do *Kinect*.

A partir da câmera RGB e do sensor de profundidade, o *Kinect* é capaz de realizar o rastreamento de usuários e gerar fluxos de dados que podem ser utilizados por uma aplicação através da API do Kinect. Estes fluxos disponibilizam seus dados em forma de quadros sucessivos de imagens, que são armazenados em *buffers*. Estes *buffers* são disponibilizados sequencialmente para a aplicação a uma taxa aproximada de 12 a 30 quadros por segundo. No caso das imagens obtidas através do sensor de profundidade, cada elemento de imagem (*pixel*) representa a distância cartesiana do plano da câmera até o objeto mais próximo naquela coordenada do campo de visão do sensor de profundidade. É possível configurar ambos os fluxos de dados através da API do *Kinect* para definir detalhes como a resolução das imagens geradas, o número de *buffers* onde as imagens serão armazenadas e disponibilizadas para a aplicação, etc. Estas configurações podem impactar diretamente na taxa de quadros disponibilizados por segundo para a aplicação.

O *Kinect* surgiu como um controle para jogos eletrônicos do console *XBox 360*. Em junho de 2011, foi lançada uma nova versão do sensor voltada para o sistema operacional Windows 7. Esta versão possui algumas diferenças em relação ao *Kinect* do *XBox 360*, tais como melhorias na API, um melhor suporte à conexão do *Kinect* através do uso de cabos USB e algumas novas funcionalidades na API do *Kinect*.

Existe uma importante diferença no sensor de profundidade dos dois modelos do *Kinect*. O sensor de profundidade do *Kinect* opera em uma faixa de profundidade padrão que normalmente varia de 80 centímetros a 4 metros de distância. Tudo o que estiver fora desta faixa é considerado muito perto ou muito longe para o sensor medir com precisão. Porém, o sensor *Kinect* para a plataforma Windows pode ser configurado para operar

em um modo denominado "*near mode*", que modifica a faixa dos valores de profundidade. Nesse modo, o *Kinect* passa, a medir a profundidade na faixa de valores entre 40 centímetros e 3 metros.

A utilização do *Kinect* neste trabalho será aprofundada no Capítulo 3.

3 PROPOSTA

Durante este trabalho, foi desenvolvida uma aplicação capaz de juntar os recursos de visualização volumétrica, incluindo as técnicas de recorte para exploração interna de volumes, aos inovadores dispositivos de entrada já discutidos — *WiiMote* e *Kinect* — com a finalidade de facilitar as tarefas de interação do usuário final com a aplicação.

A aplicação desenvolvida trata-se de um ambiente de visualização e exploração de dados volumétricos, que provê recursos para permitir a interação através dos dispositivos de entrada *WiiMote* e *Kinect*, além dos tradicionais dispositivos de entrada teclado e *mouse*. Mais especificamente, os dados volumétricos usados foram dados obtidos através de exames médicos, tais como a Ressonância Magnética e a Tomografia Computadorizada. Esses dados são armazenados em imagens no formato DICOM (*Digital Imaging Communications in Medicine*), que é um padrão para manipulação, armazenamento e transmissão bastante difundido para imagens médicas.

Neste capítulo, discutiremos alguns tópicos relacionados à implementação e à arquitetura do ambiente de visualização desenvolvido.

3.1 Ferramentas utilizadas

Ambientes de visualização de dados volumétricos tendem a ser bastante complexos. Nesse contexto, o uso de ferramentas e bibliotecas para facilitar o desenvolvimento da aplicação torna-se necessário, por diminuir consideravelmente a complexidade através da reutilização de códigos escritos, mantidos e testados diversas vezes por grandes comunidades de usuários, que oferecem suporte e atualizações periódicas aos seus produtos. Em termos de desenvolvimento no presente trabalho, o foco foi nas ferramentas de recorte e recursos de interação com o usuário. Para manter o foco nesses recursos, foram utilizadas algumas ferramentas e bibliotecas de terceiros para auxiliar em outras funcionalidades da aplicação.

Primeiramente, foi escolhida uma biblioteca de visualização capaz de abstrair ao máximo os processos de visualização e reconstrução das imagens médicas. Para o processo de visualização de imagens de dados volumétricos, a biblioteca VTK (*Visualization*

ToolKit (VTK, 2012)) foi selecionada, por dar suporte à visualização de imagens em geral, tanto bidimensionais quanto tridimensionais, incluindo imagens volumétricas. A biblioteca VTK permite o carregamento de arquivos DICOM e a exibição de seus dados, além da criação de um mundo virtual onde podemos inserir diversos tipos de objetos tridimensionais, misturando-os em meio aos elementos volumétricos no mundo virtual. Isto é importante, pois permite que possamos colocar em uma mesma cena tanto as imagens volumétricas construídas através dos arquivos de exames médicos quanto os cursores necessários para que o usuário guie as ferramentas de recorte.

Logo foram notados alguns problemas em relação ao carregamento de arquivos DICOM através da biblioteca VTK. Segundo a documentação da biblioteca, a leitura de arquivos DICOM através do VTK é limitada, e por isso é recomendável utilizar bibliotecas mais especializadas para esta finalidade. Por este motivo, a biblioteca ITK (*Insight Segmentation and Registration Toolkit* (ITK, 2012)) foi selecionada, pois possui um suporte maior ao carregamento de arquivos DICOM mais complexos. De fato, a biblioteca ITK se utiliza internamente de uma outra biblioteca, chamada GDCM (*Grassroots DICOM* (GDCM, 2012)), que é uma implementação de código aberto do padrão DICOM. O que levou à escolha da biblioteca ITK foi a sua forte integração com a biblioteca VTK, visto que ambas são mantidas por uma mesma empresa, chamada *Kitware*. Imagens carregadas através da biblioteca ITK, incluindo aquelas obtidas através de arquivos DICOM, podem ser utilizadas pelo VTK através de métodos de conversão disponíveis entre as duas bibliotecas.

Para ter uma integração efetivamente amigável com o usuário, é necessário utilizar uma interface de usuário o mais similar possível em relação à interface que o usuário já está acostumado a utilizar. Tendo isto em mente, a biblioteca wxWidgets (WXWIDGETS, 2012) foi escolhida para facilitar a criação de interfaces gráficas com uma aparência proxima às interfaces do sistema operacional do usuário. Dentre outros recursos, através da wxWidgets é possível criar interfaces gráficas de usuário que contenham janelas, menus, botões e diversos outros elementos que um usuário comum espera encontrar na maioria dos aplicativos em um computador pessoal.

As bibliotecas wxWidgets e VTK funcionam de forma separada: inicialmente, não existe uma forma oficial de integrar o ambiente de visualização gerado pelo VTK com uma janela criada através da wxWidgets. Porém, a wxWidgets fornece maneiras de criar componentes gráficos customizáveis para incorporar às janelas de uma aplicação. É fácil encontrar implementações customizadas criadas por membros da comunidade VTK que

permitem a integração transparente entre as bibliotecas VTK e wxWidgets, permitindo que a biblioteca VTK gere visualizações que podem ser exibidas como um componente dentro de uma janela criada pela wxWidgets. No presente trabalho, foi utilizada uma destas implementações para integrar as duas bibliotecas e fornecer um ambiente de trabalho mais confortável ao usuário final.

Para a utilização do *WiiMote*, foram levantadas algumas possibilidades de bibliotecas disponíveis. Não existe uma biblioteca oficial disponibilizada para o público para utilizar o controle da *Nintendo*, e todas as bibliotecas não-oficiais existentes surgiram a partir de processos de engenharia reversa sobre os protocolos de comunicação do controle. Estes fatores tornam a escolha da biblioteca uma etapa difícil, pois o suporte de bibliotecas não-oficiais é escasso e, por diversas vezes, não confiável. Um outro fator importante que torna difícil esta escolha é a documentação, muitas vezes inexistente ou embutida no próprio código fonte destas bibliotecas. A biblioteca escolhida para a captura dos dados do *WiiMote* foi a *WiiYourself!* (WIIYOURSELF!, 2012), principalmente por oferecer melhor suporte ao uso do acessório *Wii Motion Plus*, para o controle *WiiMote*. O *Wii Motion Plus* é um acessório que pode ser acoplado ao controle *WiiMote* para melhorar a precisão do acelerômetro do controle, o que é interessante para a aplicação desenvolvida neste trabalho.

Por último, a biblioteca oficial da Microsoft para a utilização do *Kinect*, conhecida simplesmente por *Kinect SDK*, foi escolhida para a utilização do sensor na aplicação. O sensor utilizado durante os testes do projeto foi um sensor *Kinect* para *XBox*.

A linguagem C++ foi utilizada para o desenvolvimento do aplicativo, trazendo como principais benefícios a sua capacidade de uso do paradigma de orientação a objetos e suas características de linguagem de alta performance, bastante importantes para aplicações interativas.

3.2 Detalhes da implementação e arquitetura

Do ponto de vista da implementação da aplicação, foram definidos dois objetivos principais: o código da aplicação deveria ser desenvolvido de forma que permitisse uma flexibilidade para adicionar novos dispositivos de interação com o usuário, assim como novas ferramentas de recorte, além daquelas que foram inicialmente definidas.

Inicialmente, foi criada a classe **CApp**. Esta classe é responsável por gerenciar os

principais recursos da aplicação e controlar o fluxo de execução da mesma. Ela segue uma implementação em um padrão *Singleton* (FREEMAN; FREEMAN, 2009): a qualquer momento durante a execução da aplicação, existirá, no máximo, uma única instância desta classe. Esta instância é criada logo que a aplicação inicia, e possui métodos que são executados durante a inicialização do programa, para inicializar seus principais recursos (criar a janela principal da aplicação, inicializar seus diversos módulos, inicializar bibliotecas, etc.), e métodos que são executados durante a finalização, para liberar quaisquer recursos alocados e executar os procedimentos necessários para fechar a aplicação corretamente.

Para que houvesse flexibilidade em termos de dispositivos de entrada no aplicativo, foi criada a classe **CInputDeviceProcessor**. A finalidade desta classe é criar uma interface comum que deve ser obedecida pelas diversas implementações que tratam diferentes dispositivos de entrada da aplicação. A Figura 3.1 ilustra, de maneira simplificada, os principais métodos da interface definida pela da classe **CInputDeviceProcessor**, bem como a hierarquia contendo as classes especializadas **CWiiMoteInputProcessor** e **CKinectInputProcessor**. Estas últimas são as implementações responsáveis pelo tratamento dos dispositivos de entrada *WiiMote* e *Kinect*, respectivamente. A instância da classe **CApp** mantém uma referência a um objeto da classe **CInputDeviceProcessor**, sem se importar com o dispositivo de entrada específico que é tratado por este objeto. Desta forma, a classe **CInputDeviceProcessor** fornece uma camada de abstração que comunica a aplicação com os dispositivos de entrada oferecidos para o usuário.

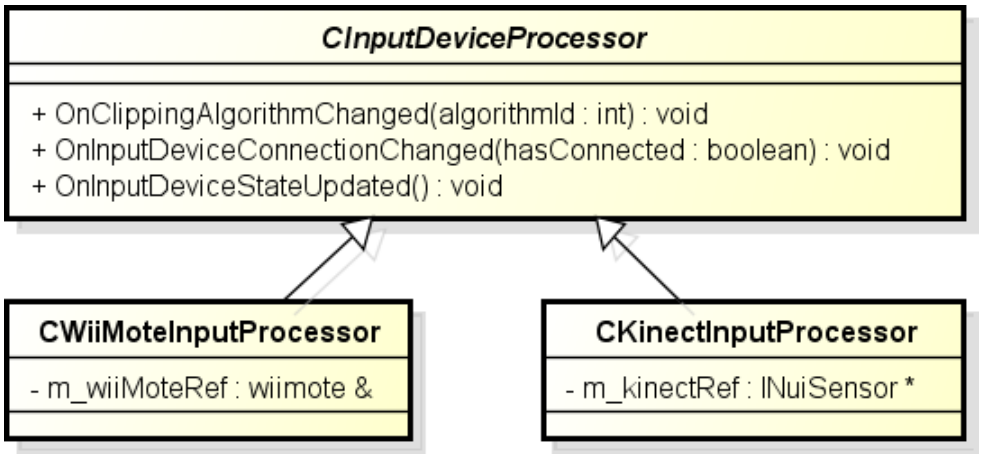

Figura 3.1: Métodos principais e hierarquia da classe CInputDeviceProcessor.

Para satisfazer o objetivo de tornar a aplicação flexível na hora de adicionar novas ferramentas de recorte, foi projetada uma outra solução também baseada em uma interface comum fornecida por uma superclasse. Neste caso, foi criada a classe **CClippingAl-**

gorithm, que representa a generalização de um algoritmo de recorte volumétrico. Cada algoritmo disponível na aplicação é implementado como uma subclasse de **CClippingAlgorithm**, conforme ilustrado na Figura 3.2.

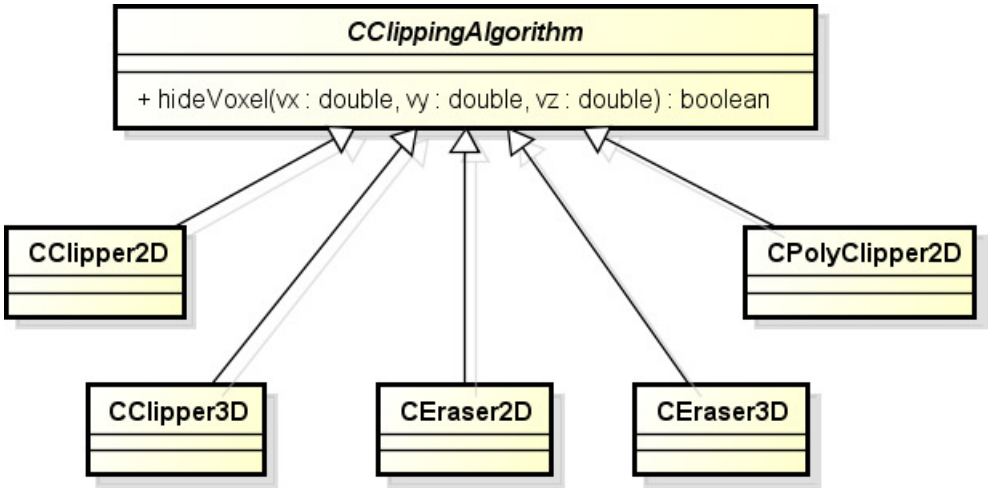

Figura 3.2: Métodos principais e hierarquia da classe CClippingAlgorithm.

Durante a execução da aplicação, quando o usuário executa uma operação de recorte, a aplicação imediatamente percorre todos os *voxels* do volume que está sendo visualizado. Para cada *voxel*, é estimada sua posição dentro do mundo virtual. Esta posição é passada para o método **hideVoxel** do objeto **CClippingAlgorithm** que representa o algoritmo de recorte que foi selecionado pelo usuário. A implementação de cada ferramenta de recorte consiste em criar uma classe específica para a ferramenta de recorte, que seja uma especialização da classe **CClippingAlgorithm**, e sobrescrer o método **hideVoxel**, de forma que, ao receber a posição estimada para um *voxel* qualquer no mundo virtual, a ferramenta de recorte deverá determinar se este *voxel* deve ser selecionado ou não. Isto é feito através do retorno de um valor verdadeiro ou falso pelo método **hideVoxel** da classe **CClippingAlgorithm**. Um valor verdadeiro indica que a ferramenta selecionou aquele *voxel* para ser removido da visualização, enquanto que um valor falso indica à aplicação que a ferramenta não irá afetar o *voxel* que está naquela posição.

A classe **CApp** mantém uma referência para um objeto **CClippingAlgorithm** que representa a técnica de recorte que foi selecionada pelo usuário na aplicação. Isto constitui um padrão de projeto chamado *Strategy* (FREEMAN; FREEMAN, 2009). Neste padrão, é definida uma família de algoritmos, que são encapsulados e são intercambiáveis. No caso da aplicação desenvolvida, os algoritmos usados pelas técnicas de recorte são encapsulados em classes, que representam seus comportamentos, e são trocados toda vez que o usuário seleciona uma nova técnica de recorte na aplicação.

3.3 Uso da aplicação

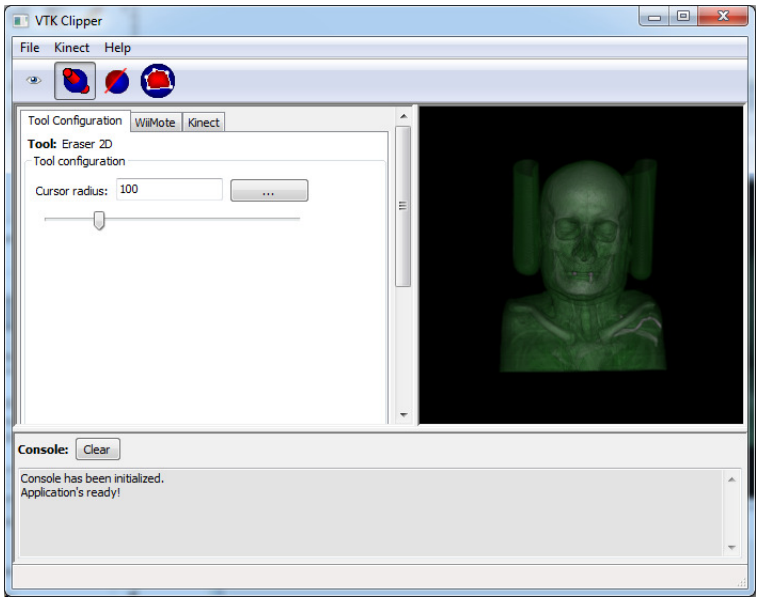

Figura 3.3: Aplicação desenvolvida em execução.

Em um cenário normal da utilização do ambiente de visualização desenvolvido, o usuário utiliza o menu da aplicação para selecionar uma pasta contendo arquivos no formato DICOM com os dados volumétricos de um exame médico a serem carregados. A aplicação irá, então, carregar os dados e exibir uma representação tridimensional construída a partir desses dados. Neste momento, o usuário tem a sua disposição uma barra de ferramenta contendo ícones, que representam as ferramentas de recorte volumétrico que podem ser selecionadas. Nesta mesma barra também existe uma ferramenta especial de visualização, ilustrada com uma imagem de um olho, que permite que o usuário altere alguns parâmetros da visualização, como a distância de visualização em relação aos dados e a orientação da câmera. A Figura 3.3 ilustra a aplicação em execução.

Algumas vezes é interessante ter a liberdade de configurar certos parâmetros de algumas ferramentas de recorte. Para as ferramentas da Borracha 2D e Borracha 3D, por exemplo, pode ser interessante permitir que o usuário configure o raio da circunferência ou do cilindro que serve de cursor para estas ferramentas. Uma outra funcionalidade interessante para a aplicação seria permitir que o usuário também pudesse controlar certas configurações dos dispositivos de entrada na aplicação. Para estas finalidades, existe, ao lado esquerdo da área de visualização da aplicação, um painel dividido em três abas: uma aba *Tool Configuration*, para configuração de parâmetros da ferramenta de recorte atualmente selecionada e as abas *WiiMote* e *Kinect*, onde o usuário pode configurar algumas opções desses dispositivos de entrada.

O usuário pode interagir com a aplicação através do uso de dispositivos tradicionais (teclado e *mouse*), do *WiiMote* e do *Kinect*. A ativação dos dispositivos *Kinect* e *WiiMote* na aplicação é multuamente exclusiva, ou seja, apenas um dos dois dispositivos pode estar ativo por vez. Para ativar um deles, o usuário deve ir na aba de configuração do dispositivo de interesse e clicar no seu ícone. Isto iniciará um processo de estabelecimento de conexão com o dispositivo escolhido, que depende do dispositivo. O usuário também pode optar por desativar ou desconectar o dispositivo de entrada, clicando novamente no mesmo botão que usou para estabelecer a conexão com o dispositivo.

O uso do *mouse* e teclado está limitado às ferramentas que utilizam cursores bidimensionais: Borracha 2D, Guilhotina 2D e Recorte Poligonal 2D. Analogamente, os usos do *WiiMote* e do *Kinect* estão limitados às ferramentas que utilizam cursores tridimensionais: Borracha 3D e Guilhotina 3D. Para todas as ferramentas 2D, os cursores utilizados são posicionados sobre o plano de visualização. Já no caso das ferramentas 3D, os cursores são formas geométricas tridimensionais que podem ser controladas pelo usuário no mundo virtual, através dos dispositivos de entrada. O usuário é capaz de controlar o posicionamento e a orientação destes cursores, para indicar como os recortes volumétricos devem ser feitos pela aplicação.

Na ferramenta da Borracha 2D, o usuário utiliza o *mouse* para controlar um cursor representado por uma circunferência. Um clique com o *mouse* executa a operação de recorte. O cursor em forma de circunferência, na verdade, trata-se da representação da base de um cilindro, que está alinhada com o plano de visualização. O eixo desse cilindro é uma reta perpendicular ao plano de visualização. Durante a operação de recorte utilizando a Borracha 2D, a posição de cada um dos *voxels* do volume é verificada, de forma que os *voxels* selecionados para serem removidos são aqueles contidos dentro do volume do cilindro. Para esta operação, consideramos que o cilindro possui altura infinita. A seleção dos *voxels* é feita através da verificação da distância do ponto que representa a posição de cada *voxel* até o eixo do cilindro, de forma que aqueles *voxels* que se encontram a uma distância menor que o raio do cilíndro, em relação ao eixo do mesmo, serão selecionados para remoção. A Figura 3.4 ilustra a utilização da Borracha 2D. A Listagem 3.1 exibe o trecho de código responsável por decidir se um voxel deve ou não ser removido da visualização, utilizado tanto para a Borracha 2D quanto para a Borracha 3D.

Código 3.1: Código de remoção de *voxels* para as ferramentas Borracha 2D e Borracha 3D.

```
/* Considerando:
```

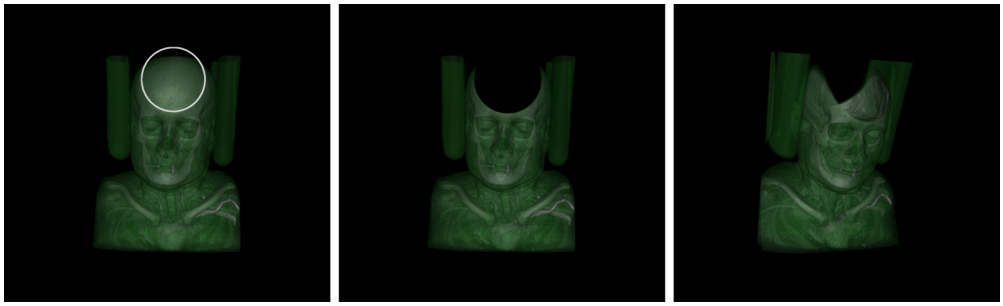

Figura 3.4: Uso da ferramenta de recorte Borracha 2D.

```
 *    voxelPos:
 *       Vetor representando a posicao do voxel, em
 *       coordenadas globais.
 *    cylinderAxis:
 *       Membro da classe CEraser, e um vetor contendo dois
 *       pontos (p0 e p1), que formam a linha do cilindro.
 *       Seu valor e configurado pelo usuario atraves do uso
 *       do cursor (posicionamento e orientacao do cilindro
 *       de recorte).
 *    cylinderRadius:
 *       Float representando o raio do cilindro de recorte.
 */

bool CEraser::hideVoxel( const Vector3 &voxelPos )
{
  // Calcular a projecao de "voxelPos"
  // em "cylinderAxis"
  Vector3 v = cylinderAxis.p1 - cylinderAxis.p0;
  Vector3 w = voxelPos - cylinderAxis.p0;

  double c1 = Vector3::dotProduct(w,v);
  double c2 = Vector3::dotProduct(v,v);
  double b = c1 / c2;

  Vector3 projectedPoint = cylinderAxis.p0 + b * v;

  // Remover o voxel da visualizacao, caso a distancia
```

```
  // do mesmo ao eixo do cilindro de recorte seja menor
  // que o raio do cilindro (ou seja, se o voxel se
  // encontra na regiao interna do cilindro).
  return (
    Vector3::distance(voxelPos, projectedPoint) <
    cylinderRadius
  );
}
```

Para utilizar a ferramenta Guilhotina 2D, o usuário usa o *mouse* para demarcar dois pontos no plano de visualização, através de cliques. Estes dois pontos definem uma reta, que representa um plano perpendicular ao plano de visualização. Em seguida, o usuário deve definir um terceiro ponto, que definirá qual região do volume será removida da visualização. Através de operações de produto escalar entre a posição de um ponto e um vetor normal a um plano, podemos obter uma indicação sobre a posição do ponto em relação ao plano: o resultado desta operação será um resultado positivo de um lado do plano, ou negativo do outro lado do plano. Tendo isso em mente, podemos descobrir e remover os *voxels* que se situam do mesmo lado do plano que foi escolhido pelo usuário. A Figura 3.5 ilustra a utilização da Guilhotina 2D. A Listagem 3.2 exibe o trecho de código responsável por decidir se um voxel deve ou não ser removido da visualização, utilizado tanto para a Guilhotina 2D quanto para a Guilhotina 3D.

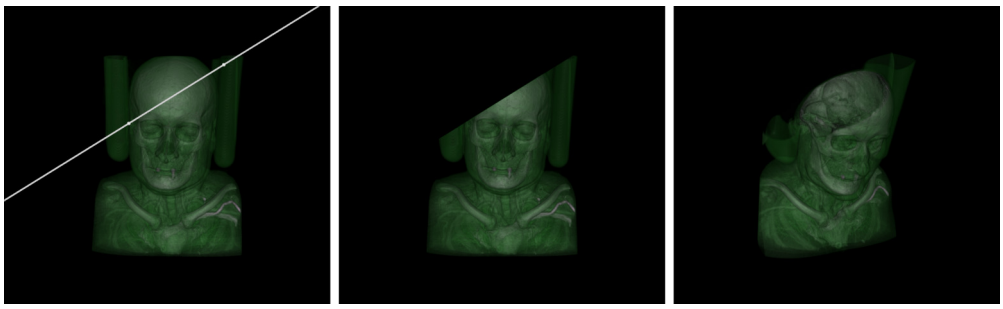

Figura 3.5: Uso da ferramenta de recorte Guilhotina 2D.

Código 3.2: Código de remoção de *voxels* para as ferramentas Guilhotina 2D e Guilhotina 3D.

```
/* Considerando:
 *    voxelPos:
 *       Vetor representando a posicao do voxel, em
 *       coordenadas globais.
```

```
 *    pointOnPlane:
 *       Um ponto qualquer contido no plano de recorte.
 *       Configurado pelo usuario, atraves do posicionamento
 *       do cursor da ferramenta de recorte.
 *       Membro da classe CClipper.
 *    planeNormal:
 *       A normal do plano de recorte. Tambem configurada
 *       pelo usuario atraves do cursor. Podemos definir o
 *       plano de recorte usando um ponto (pointOnPlane) e
 *       a normal do plano (planeNormal).
 *       Membro da classe CClipper.
 *    erasePositives:
 *       Uma flag indicando se pontos do lado positivo do
 *       plano devem ser apagados (true), ou se pontos do
 *       lado negativo devem ser apagados (false).
 *       Membro da classe CClipper.
 */
bool CClipper::hideVoxel( const Vector3 &voxelPos )
{
  double sign = Vector3::dotProduct( planeNormal, voxelPos -
    pointOnPlane );
  return (erasePositives == (sign > 0));
}
```

A última das ferramentas 2D é o Recorte Poligonal 2D. Nesta ferramenta, o usuário utiliza o *mouse* para definir vários pontos no plano de visualização, através de cliques. Estes pontos formam um polígono que define uma área de seleção dos voxels a serem removidos. Após definir um polígono, o usuário deve segurar uma tecla especial do teclado (tecla *CTRL*) e clicar na região interna ou na região externa do polígono definido. Feito isto, a aplicação irá dar início ao recorte volumétrico. Para esta ferramenta, os voxels são projetados no plano de visualização e um algoritmo é usado para selecionar aqueles *voxels* cuja projeção está na mesma região selecionada pelo usuário em relação ao polígono definido: a região interna ou a região externa a este polígono. A Figura 3.6 ilustra a utilização do Recorte Poligonal 2D. A Listagem 3.3 exibe o trecho de código responsável por decidir se um voxel deve ou não ser removido da visualização, utilizado para o Recorte

Poligonal 2D.

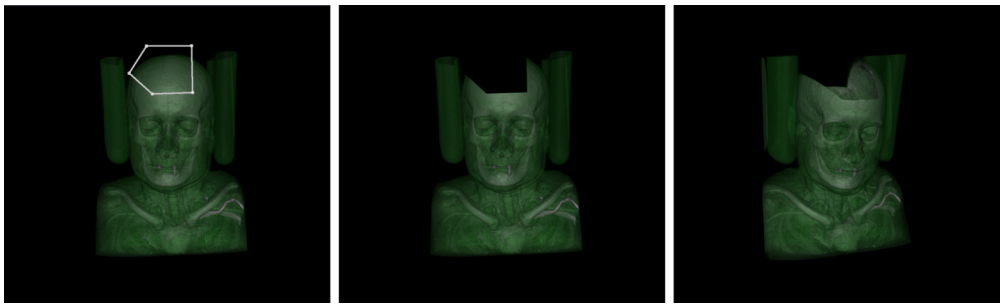

Figura 3.6: Uso da ferramenta de Recorte Poligonal 2D.

Código 3.3: Código de remoção de *voxels* para a ferramenta Recorte Poligonal 2D.

```
/* Considerando:
 *    voxelPos:
 *       Vetor representando a posicao do voxel, em
 *       coordenadas globais.
 *    eraseInsidePoly:
 *       Flag indicando se os voxels a serem removidos
 *       sao os que estao dentro (true) ou fora (false)
 *       do poligono.
 *    isPointInsidePoly:
 *       Um metodo usado para verificar se um ponto esta
 *       contido dentro de um dado poligono convexo. O
 *       poligono em questao seria aquele definido pelo
 *       usuario, atraves de cliques com o mouse.
 */
bool CPolyClipper2D::hideVoxel( const Vector3 &voxelPos )
{
  // Verifica se os voxels estao dentro ou fora
  // do poligono.
  bool inPoly = isPointInsidePoly( voxelPos );
  return (inPoly == eraseInsidePoly);
}
```

As ferramentas 3D implementadas na aplicação funcionam a partir de três passos básicos: configuração dos parâmetros específicos da ferramenta selecionada, posicionamento do cursor e orientação do mesmo. Após esses três passos básicos, o processo de

recorte é iniciado para a remoção de *voxels*. Nesse contexto, os dispositivos de entrada *WiiMote* e *Kinect* são utilizados pelo usuário para alterar a posição e a orientação do cursor da ferramenta de recorte 3D atualmente selecionada na aplicação.

Na ferramenta da Borracha 3D, o usuário deve controlar um cursor cilíndrico no mundo virtual. O cilindro deve ser posicionado e rotacionado pelo usuário, e os *voxels* que serão selecionados para o recorte serão aqueles que estão contidos na região interna do cilindro. Isto é feito da mesma forma que a ferramenta Borracha 2D, calculando-se a distância da posição do *voxel* até o eixo do cilindro, e comparando-se esta distância com o raio do cilindro. A Figura 3.7 ilustra a utilização da Borracha 3D.

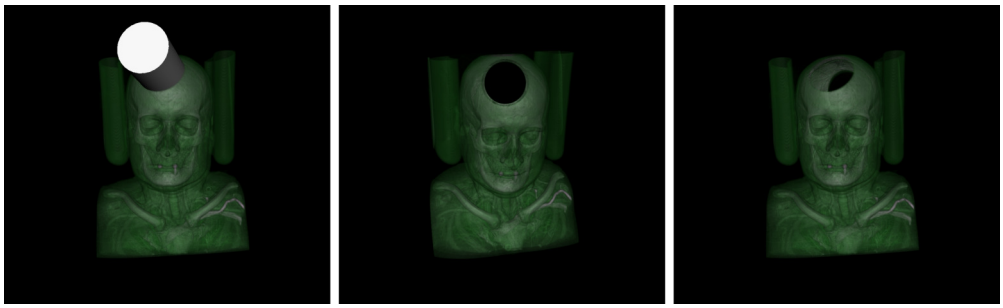

Figura 3.7: Uso da ferramenta de recorte Borracha 3D.

Na ferramenta da Guilhotina 3D, o usuário deve controlar um cursor que tem a forma de um polígono quadrilátero e representa o plano de recorte da guilhotina. Partindo do ponto central do quadrilátero, existe uma linha vermelha, perpendicular ao plano de recorte, que indica o lado do plano que será removido quando for executada a operação de recorte. O quadrilátero deve ser posicionado e rotacionado pelo usuário, e os *voxels* que serão selecionados para o recorte serão aqueles que estão contidos na região que fica do lado do plano que contém a linha vermelha. Isto é feito através de um cálculo baseado em operações de produto escalar, da mesma forma utilizada na Guilhotina 2D. A Figura 3.8 ilustra a utilização da Guilhotina 3D.

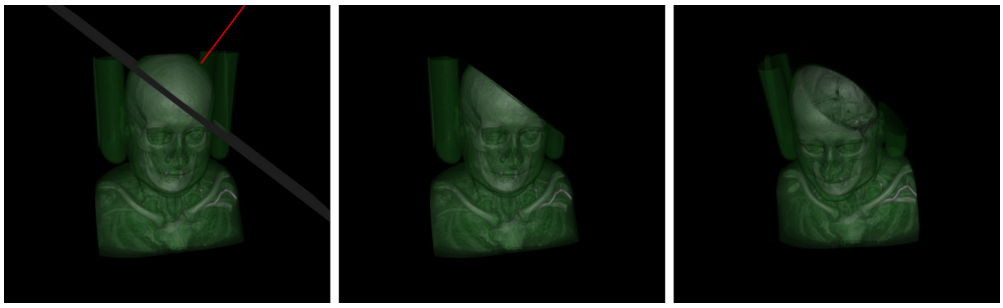

Figura 3.8: Uso da ferramenta de recorte Guilhotina 3D.

A utilização do *WiiMote* para o controle do posicionamento e da orientação dos curso-

res das ferramentas de recorte 3D funciona de duas formas, dependendo das configurações feitas pelo usuário na aplicação. A primeira é baseada na orientação do acelerômetro. O usuário deve segurar o controle horizontalmente, e girar o mesmo para modificar a posição ou a orientação do cursor. A segunda forma de uso do controle é baseada na medição das variações da aceleração do *WiiMote*. Nesse caso, o usuário deve segurar o *WiiMote* verticalmente, como um controle remoto, e movimentá-lo para os lados, para cima ou para baixo. Isto gera uma aceleração que é medida pelo acelerômetro do *WiiMote* e usada para posicionar ou modificar a orientação do cursor da ferramenta de recorte.

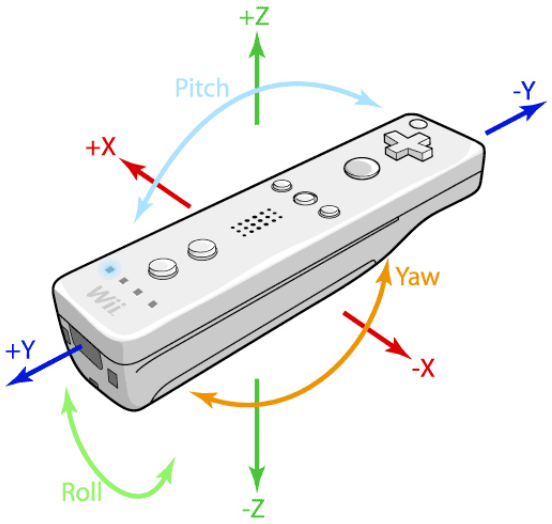

Figura 3.9: Eixos do *WiiMote*.

Durante a utilização do *WiiMote*, podemos obter os seus valores de aceleração nos treis eixos: x, y e z (ilustrados na Figura 3.9. Porém, ao monitorar os dados obtidos através das medições do acelerômetro, notam-se pequenas variações, mesmo com o controle totalmente parado e estático sobre uma superfície plana. Estas variações ocorrem por conta da precisão das medições do acelerômetro e de sua calibração, e podem causar comportamentos indesejáveis durante a interação com o usuário da aplicação. Outro ponto importante a ser considerado, do ponto de vista do usuário, é que deve haver uma forma de parar a interação e retomá-la a qualquer momento, pois pode ser incomodo para o usuário não ter a opção de interromper o que está fazendo caso precise.

Uma forma encontrada para resolver estes dois problemas foi utilizar um botão do *WiiMote* que ativa ou desativa a funcionalidade do acelerômetro, permitindo que a leitura dos valores de aceleração seja interrompida a qualquer momento. O botão traseiro do *WiiMote* foi utilizado para isso, de forma que a leitura dos valores de aceleração e a interação com o usuário só ocorre efetivamente quando o botão está pressionado. Foi implementada também uma outra forma de amortizar as variações do acelerômetro e

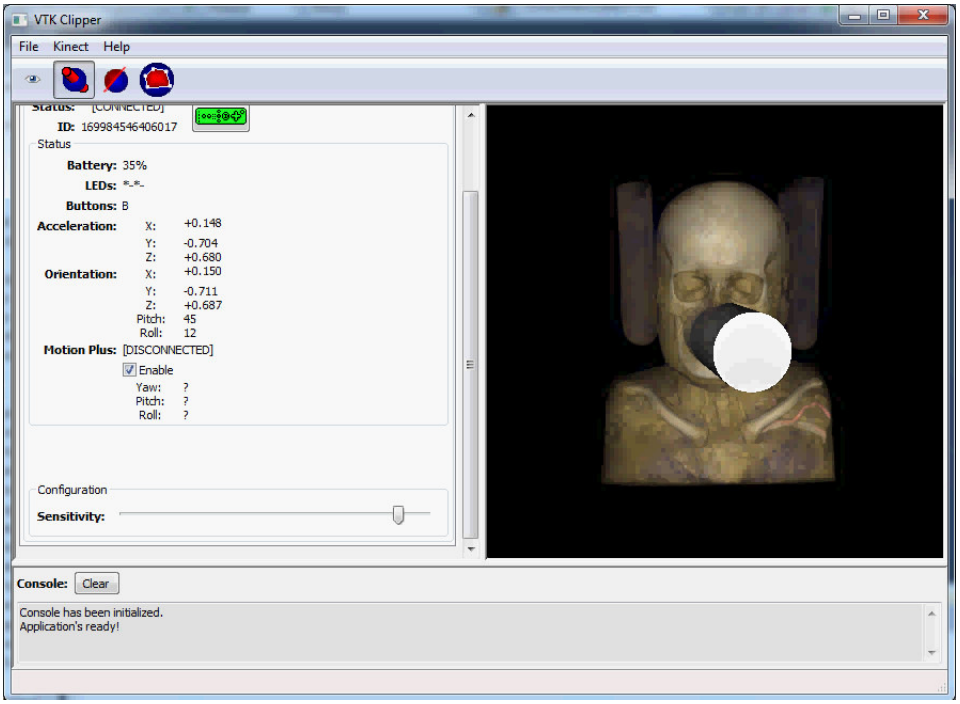

Figura 3.10: Painel de configurações e monitoramento do *WiiMote*.

permitir um melhor controle da movimentação dos cursores pelo usuário, baseada em um fator de sensibilidade que pode ser configurado através da interface da aplicação a qualquer momento. Assim, o usuário pode configurar o quanto a movimentação do controle irá afetar o cursor da ferramenta 3D.

Já a utilização do *Kinect* na aplicação baseia-se em menus virtuais, que o usuário pode "tocar", utilizando as mãos. No painel de configurações do *Kinect*, existe uma pequena tela onde são exibidas as silhuetas dos usuários identificados pelo sensor e um menu de opções. Cada usuário identificado possui uma cor diferente nesta tela, para diferenciá-los, e o menu de opções é desenhado de forma que sobrepõe as silhuetas dos usuários. Apenas um dos usuários identificados é capaz de executar as operações de recorte do *Kinect*. Este usuário utiliza as mãos para tocar nos botões do menu virtual e interagir com os cursores das ferramentas de recorte 3D. De fato, existem vários menus na aplicação, e não apenas um. Cada menu é especializado em uma parte da operação de recorte: existe um menu para selecionar a ferramenta de recorte desejada, um menu para alterar a posição do cursor de recorte e um menu para alterar a orientação do cursor. Em cada menu, existe também um botão de confirmação, usado para avançar para o menu seguinte, e um botão de retrocesso, usado para voltar para o menu anterior. A Figura 3.11 ilustra a utilização do Kinect por parte do usuário da aplicação.

A aplicação possui um sistema com vários menus, de forma que o usuário pode navegar entre eles através do toque nos botões dos menus. Inicialmente, a aplicação apresenta um

menu que permite ao usuário escolher uma das ferramentas de recorte. Após tocar um dos botões que indica uma das ferramentas de recorte, esta ferramenta é selecionada e seu cursor 3D é exibido para o usuário. Então, o menu de seleção da ferramenta de recorte é escondido, e o menu de posicionamento do cursor é exibido. Neste menu, o usuário vê três botões para escolher um dos três eixos do espaço onde ele deseja movimentar o cursor, e mais dois botões indicados por um sinal positivo e um sinal negativo. O usuário deve selecionar um dos eixos e modificar a posição do cursor através do incremento (botão com sinal positivo) ou decremento (botão com sinal negativo) de sua coordenada no eixo selecionado. Após ajustar a posição do cursor, o usuário pode tocar o botão de confirmação, que o levará para o próximo menu: o menu de rotação do cursor. Neste menu, o usuário pode alterar a orientação do cursor através de um sistema similar ao menu de posicionamento do cursor: existem três botões para escolha de um eixo e um botão para incremento e outro para decremento do ângulo de rotação ao redor do eixo selecionado. Ao tocar o botão de confirmação, a aplicação inicia o recorte do volume.

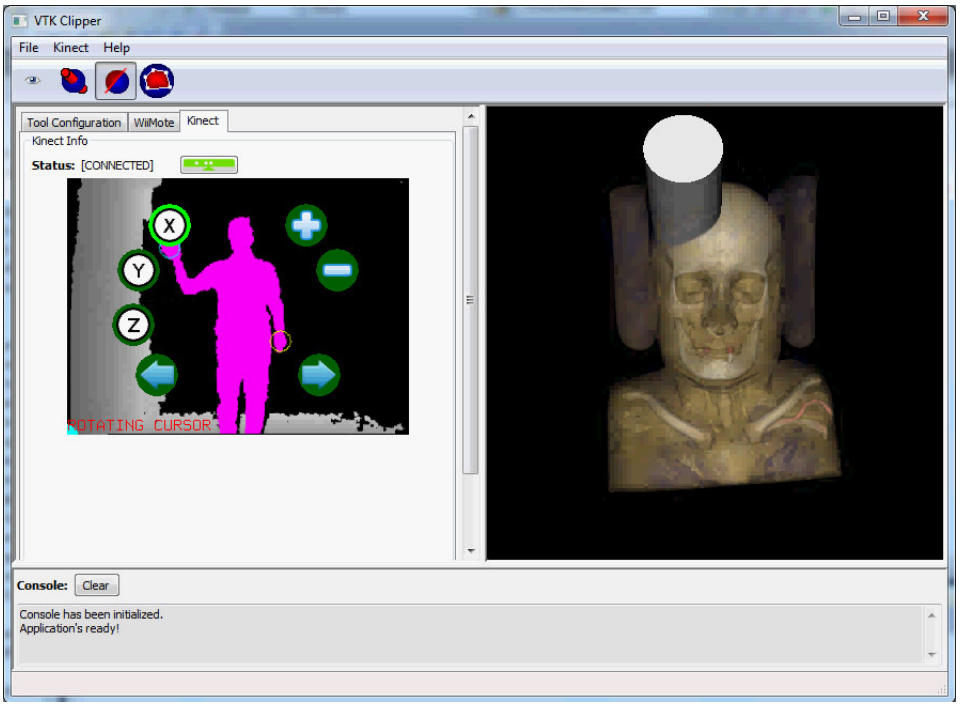

Figura 3.11: Painel de configurações e utilização do *Kinect*.

Para avaliar a performance dos algoritmos de recorte, foi realizado um experimento em uma máquina com processador Intel Core i5-760, 4GB de memória RAM e placa de vídeo NVidia GeForce GTX 750, rodando o sistema operacional Windows 7 Ultimate SP1. Os testes foram realizados utilizando dados de uma série de 460 imagens no formato DICOM, cada uma com resolução de 512 *pixels* de largura por 512 *pixels* de altura, obtidas a partir de um exame de Tomografia Computadorizada. O experimento consistiu em 10 medições

dos tempos de recorte das ferramentas da aplicação, para obter uma média aritmética dos tempos medidos. A Tabela 3.1 indica os resultados de desempenho da aplicação, em termos das médias de tempo gasto para processar cada ferramenta de recorte sobre os dados volumétricos do teste.

Ferramenta	**Tempo médio**
Borracha 2D	4,22s
Guilhotina 2D	1,57s
Recorte Poligonal 2D[1]	16,55 s
Borracha 3D	3,21 s
Guilhotina 3D	1,58 s

Tabela 3.1: Desempenho nos testes da aplicação.

[1]Foram utilizados hexágonos durante os testes.

4 CONCLUSÃO

A visualização volumétrica vem se tornando cada vez mais importante para a medicina. Exames médicos modernos são capazes de gerar conjuntos complexos de dados que, em geral, precisam ser analisados cuidadosamente por médicos e especialistas. Portanto, é muito importante que os *softwares* de visualização volumétrica utilizados durante a análise desses conjuntos complexos de dados possuam recursos que garantam uma interação simples e intuitiva aos seus usuários, aumentando seu grau de imersão durante suas tarefas. A forma como o usuário interage com a aplicação pode causar um grande impacto na sua produtividade, gerando resultados mais positivos, como, por exemplo, a diminuição da probabilidade de erro ao analisar os dados contidos em uma imagem médica. No presente trabalho, foi desenvolvido um ambiente de visualização de dados volumétricos, focado em imagens médicas, capaz de satisfazer esta necessidade de explorar diferentes maneiras de interação do usuário com o *software* desenvolvido.

A arquitetura proposta para facilitar a inclusão de novas ferramentas de recorte e novos dispositivos de entrada provou ser bastante flexível, para ambos os casos. Isto se deve principalmente à adoção de alguns padrões de desenvolvimento de *software* empregados durante a implementação do projeto. Por outro lado, algumas dificuldades de implementação surgiram em decorrência da utilização do controle *WiiMote* no projeto, pois as bibliotecas encontradas não são oficialmente suportadas pela *Nintendo*, possuindo, em geral, uma documentação escassa e apresentando algumas limitações.

Em relação às ferramentas de recorte implementadas no presente trabalho, uma atenção especial foi dada aos tempos de execução de cada ferramenta. Durante o desenvolvimento, foram feitas diversas alterações nas implementações de cada ferramenta, com o objetivo de otimizar seus tempos de execução. Isto foi feito porque para o usuário final, pode ser bastante incomodo ter que esperar um tempo muito grande para executar um recorte no volume que está sendo visualizado. Como resultado final, foram obtidos tempos aceitáveis para a execução de cada ferramenta de recorte. Para trabalhos futuros em relação às ferramentas disponibilizadas, podem ser implementadas novas ferramentas, já que existem bastantes ferramentas descritas na literatura. Outra melhoria para a aplicação seria a implementação destas ferramentas de recorte diretamente na placa gráfica, em oposição à atual implementação, que foi feita via *software*. A implementação

de ferramentas de recorte diretamente na placa gráfica pode ser vista em outros trabalhos presentes na literatura (HUFF, 2006).

O uso do *WiiMote* inicialmente não é completamente intuitivo para o usuário, e precisa de alguma prática inicial para se tornar simples. Após esta etapa de prática, é possível adquirir o costume de usar o *WiiMote* para realizar os procedimentos de recorte de maneira confortável. Um problema que pode ser destacado da utilização do *WiiMote* é a baixa precisão oferecida por seu acelerômetro, que deixa muito a desejar em termos de sensibilidade. Este problema pode ser amenizado através do uso de um acessório disponível para o controle, chamado *Wii Motion Plus*. Este é um acessório oficialmente disponibilizado pela *Nintendo* capaz de melhorar a sensibilidade do acelerômetro do *WiiMote*. Como trabalhos futuros relacionados à utilização do *WiiMote*, pode ser sugerida a utilização das suas capacidades de dispositivo apontador, para a utilização das ferramentas de recorte 2D. Uma outra sugestão seria o teste de utilização de outras bibliotecas para a comunicação com o *WiiMote*, já que a biblioteca *WiiYourself!* apresenta algumas falhas notáveis e foi descontinuada em 2010.

Já a utilização do *Kinect* é mais intuitiva para o usuário. O sensor trabalha diretamente com a identificação do usuário e dos seus movimentos, removendo a necessidade do usuário ter que segurar e aprender a manusear corretamente um dispositivo físico em suas mãos, como é o caso do *WiiMote*. As capacidades de interface natural de usuário oferecidas pelo *Kinect* o tornam um dispositivo de entrada bastante atraente, permitindo uma interação imersiva do usuário, com pouca prática. Como trabalhos futuros sugeridos para o *Kinect*, podemos estender sua funcionalidade através do uso de seus recursos de processamento de áudio e reconhecimento de voz para a execução de algumas tarefas na aplicação, tais como: selecionar uma nova ferramenta de recorte, executar a operação de recorte atualmente selecionada ou selecionar um dos eixos para modificar o posicionamento ou a orientação de um cursor de ferramenta de recorte 3D. O uso de comandos de voz para estas três operações poderia ter um impacto bastante positivo na imersão do usuário.

Em relação ao ambiente de visualização em si, futuramente podem ser adicionados novos dispositivos de entrada e oferecido um melhor suporte às funções de transferência utilizadas para o *rendering* do volume, que atualmente são definidas de maneira constante no código da aplicação.

REFERÊNCIAS

ANDREJASIC, M.; POBERAJ, I. *MEMS Accelerometers*. 2008. Seminário. University of Ljubljana, Faculdade de matemática e física, Departemento de Física, Marec.

ELVINS, T. T. *A Survey of Algorithms for Volume Visualization*. [S.l.]: ACM SIGGRAPH, 1992.

FIGUEIREDO, L. J. et al. *Aplicações de Acelerómetros*. 2007. Monografia. Lisboa, Portugal, 19 Dezembro.

FREEMAN, E.; FREEMAN, E. *Use a cabeça: padrões e projetos*. STARLIN ALTA CONSULT, 2009. ISBN 9788576081746. Disponível em: <http://books.google.com.br/books?id=VRSPPgAACAAJ>.

GDCM. 2012. Disponível em http://gdcm.sourceforge.net/. Acessado em maio de 2012.

HOW Stuff Works: Wii. 2012. Disponível em http://electronics.howstuffworks.com/wii.htm. Acessado em maio de 2012.

HUFF, R. *Recorte Volumétrico Usando Técnicas de Interação 2D e 3D*. 2006. Dissertação (Pós Graduação em Computação). PPGC da UFRGS, Porto Alegre.

ITK. 2012. Disponível em http://www.itk.org/. Acessado em maio de 2012.

KINECT for Windows. 2012. Disponível em http://www.kinectforwindows.org/. Acessado em maio de 2012.

MANSSOUR, I. H. et al. Visualizing inner structures in multimodal volume data. In: *Proceedings of the 15th Brazilian Symposium on Computer Graphics and Image Processing*. Washington, DC, USA: IEEE Computer Society, 2002. (SIBGRAPI '02), p. 51–58. ISBN 0-7695-1846-X. Disponível em: <http://dl.acm.org/citation.cfm?id=646016.678122>.

MCCORMICK, B. H.; DEFANTI, T. A.; BROWN, M. D. *Visualization in scientific computing*. [S.l.]: ACM SIGGRAPH, 1987.

PAIVA, A. C. de; SEIXAS, R. de B.; GATTASS, M. *Introdução à Visualização Volumétrica*. Janeiro 1999. PUCRio Inf.MCC03/99.

RAVI, N. et al. Activity recognition from accelerometer data. In: *Proceedings of the 17th conference on Innovative applications of artificial intelligence - Volume 3*. AAAI Press, 2005. (IAAI'05), p. 1541–1546. ISBN 1-57735-236-x. Disponível em: <http://dl.acm.org/citation.cfm?id=1620092.1620107>.

SOUZA, H. de P. *Exploração de Dados Volumétricos Através de Ferramentas de Recorte*. 2010. Monografia. DEINF-UFMA.

VTK. 2012. Disponível em http://www.vtk.org/. Acessado em maio de 2012.

WIIBREW. 2012. Disponível em http://wiibrew.org/. Acessado em maio de 2012.

WIILI. 2012. Disponível em http://homepage.mac.com/ianrickard/wiimote/wiili_wimote.html. Acessado em maio de 2012.

WIIYOURSELF! 2012. Disponível em http://wiiyourself.gl.tter.org/. Acessado em maio de 2012.

WXWIDGETS. 2012. Disponível em http://www.wxwidgets.org/. Acessado em maio de 2012.